1-1-2010

General probability distribution-based QoS analysis for web service composition

Huiyuan Zheng
Macquarie University

Jian Yang
General Research Institute for Non Ferrous Metals,Ministry of Science & Technology, China,Macquarie University

Weiliang Zhao
University of Wollongong, wzhao@uow.edu.au

Probability Distribution-Based QoS Analysis for Web Service Composition

Huiyuan Zheng, Jian Yang, and Weiliang Zhao

Computing Department, Macquarie University,
NSW, 2109, Australia
{huiyuan.zheng,jian.yang,weiliang.zhao}@mq.edu.au

Abstract. In this paper, we propose a comprehensive solution for QoS aggregation in service composition. Different from existing work on QoS modeling which uses single values, discrete values with frequencies, or well known statistical distributions, the proposed method can work on QoS represented by general probability distributions. A set of formulas are developed to calculate the QoS probability distribution for the composite service based on the four identified patterns. Experimental results are provided to demonstrate the effectiveness of the proposed method.

1 Introduction

The nature of services creates the opportunity for building composite services by combining existing elementary or complex services (referred to as component services) from different enterprises and in turn offering them as high-level services or processes. QoS analysis becomes increasingly challenging and important when complex and mission critical applications are built upon services with different QoS [1]. Thus solid model and method support for QoS predication in service composition become crucial and will lay a foundation in further analysis of complexity and reliability in developing service oriented distributed applications.

It is important to estimate the QoS of a composite service at design time based on the quality of individual Web services to make sure that the composition can satisfy the expectations of end users [10,2]. A Web service may need to be replaced at run time if it becomes unavailable or its performance degrades too much [3]. Quite often, functionally equivalent services exist with different QoS. A comparison is therefore necessary by analysing the QoS of the composite service with different service combination options.

Two issues need to be addressed to perform QoS analysis: (1) how QoS of a service can be accurately represented? (2) how QoS of a composite service can be calculated based on the QoS of its component services?

Currently three types of QoS representations are used: single values, discrete values with frequencies (i.e. probability mass function), and well known statistical distributions such as normal distribution. None of the above mentioned models effectively represents QoS. For example, for two Web services with the

D.K.W. Chiu et al. (Eds.): WISE 2010 Workshops, LNCS 6724, pp. 98–111, 2011.

same response time, the one with the small variance is better. But this information cannot be captured if one or a few values are used to represent the quality. Furthermore, real service execution data shows that a distribution of a QoS metric such as response time can be of any shape, which may not fit into any well known statistical distributions.

When a single QoS metric, such as availability and execution time, of a component service is represented by single or discrete values, aggregation approach is adopted to calculate the QoS for the composite service [4,6]. The existing aggregation approaches cannot handle QoS that are represented as probability distributions. When the QoS of component services are represented by well known statistical distributions, simulation approach is applied to compute the distribution of the composite service [8]. Since simulation approach uses the well known statistical distributions instead of the real QoS distributions for component services, the calculation result cannot be accurate by nature.

Furthermore, how QoS of component services are aggregated also depends on the way the composite service is constructed. [5] provides a QoS analysis tool that can handle well known statistical distributions in limited composition patterns such as sequential, parallel. The QoS aggregation for unstructured composition patterns is not fully addressed in the current QoS literature.

In order to overcome the problems discussed above, we propose a comprehensive framework for QoS analysis and aggregation. The main idea is: (1) a composite service is regarded as being constructed based on four composition patterns, i.e. sequential, parallel, conditional, and loop; (2) QoS for component service is represented by its QoS probability distribution, on which we do not make any assumption in relation to its type or shape. That is, the QoS can be a single value, a probability mass function, or a probability distribution in any shape; (3) QoS aggregation operations for different composition patterns and different quality metrics are defined and formulas are developed for these operations. QoS calculation for a composite service becomes a matter of iteratively applying the QoS aggregation operations to these composite patterns. In the rest of the paper, we will use the term *component QoS* and *composite QoS* to refer to QoS of component service and QoS of composite service respectively. We will also use the term QoS and QoS metric interchangeably.

The remainder of the paper is organized as follows: Section 2 discusses the related work in QoS aggregation. In Section 3, patterns in a composite service are defined. Aggregation operations are defined and formulas for the aggregation operations are developed in Section 4. In Section 5, the process of QoS aggregation for a composite service is explained in detail. Experimental results are provided in Section 6. Section 7 concludes the paper.

2 Related Work

Existing work uses single values [4], probability mass functions [6], or well known statistical distributions [8] to represent component QoS.

For single value represented QoS, aggregation method [4] is proposed to calculate the composite QoS. A composition can be regarded as being composed of

composition patterns. formulas to calculate QoS for these patterns are given. But these formulas can only be applied to single values. In our previous work [11], an approach is proposed to calculate the QoS for every possible execution path of a composition and then a discrete QoS distribution is generated for the composition.

For QoS represented by discrete values with frequencies (i.e. probability mass functions) [6], the calculation method is much the same as it is for single values. The difference is that the probability of each possible QoS value of the composite service needs to be taken into account.

For the well known statistical distribution represented QoS [4,8], simulation approaches are applied to compute the composite QoS. The real QoS data of component services are fitted with standard statistical distributions.

[5] presents a tool for predicting composite QoS. Component QoS can be modeled as single values or well known statistical distributions. But this tool does not support complex patterns such as loop.

The limitations and ineffectiveness of the above mentioned work has been discussed in the previous section. In comparison with the existing work, the contributions of this paper can be summarized as follows:

1. The proposed method does not put any constraints on the representation of a component QoS, i.e., it can be in single value, discrete values with frequencies, well known statistical distribution, or any distribution regardless of its shape. This characteristic is important because the representation of different QoS metrics cannot be the same. For example, a probability distribution is appropriate to represent response time; a single value is good to represent availability; while a probability mass function can represent cost well.
2. The method can deal with commonly used composition patterns, including loop with arbitrary exit points.
3. QoS for composite service can be calculated based on real QoS distribution data obtained from service execution history, which cannot be done with any existing approaches.
4. The proposed method provides a much more general approach compared with the existing ones. As a result, the problems dealt with in the existing methods for QoS aggregation become the special cases in the proposed method.

For the clarity of the research, we make the following assumptions: (1) Each QoS metric is calculated individually without considering the correlations with other QoS metrics. For example, *cost* may be correlated to *response time*. However, we will analyse and calculate the composite QoS for *cost* and *response time* separately. (2) In the composite QoS calculation, we will not consider the cases when component services seriously affected with each other on their QoSs. This situation only happens when these component services are simultaneously invoked on the same server.

3 Modeling of a Service Composition and Composition Patterns

As we discussed early on, one of the aspects that affects composite QoS calculation is the structure of the composite service. In this section, we will define the basic composition patterns that a composite service is built upon. The modeling method of a service composition and its composition patterns have been given in [11]. Here, we give a summary of the modeling method.

A service graph is applied to model a composite service. The vertices represent the component Web services and the edges denote the transitions with transition probabilities between services. The definition of the service graph is as follows:

Definition 1. *Service Graph: Let S be the set of Web services, T be the set of transitions in a composite service, and P be the set of transition probabilities between two services linked by a transition. A Service Graph is $G = (V, A)$, where*

- *$V = S$ are the vertices of the graph;*
- *$A = T \subseteq V \times \nabla \times V \times P$ are the arcs of the graph;*
- *$\nabla = \{-, ||_{syn}, ||_{sng}\}$ are connection methods in the graph with*
  - *'$-$' denotes a sequential connection*
  - *'$||_{syn}$' denotes a concurrent-synchronized connection, i.e., a concurrent connection that will be followed by a synchronized merge (the merge is triggered by the termination of all the concurrent running branches)*
  - *'$||_{sng}$' denotes a concurrent-single connection, i.e., a concurrent connection that will be followed by a single merge (the merge is triggered by the first finished branch and all the other branches that are still running will be ignored)*
- *$\forall a_i \in A$, $a_i = (v_x \Phi v_y, p)$ where $\Phi \in \nabla$ and $p \in P$, denoting that the arc from vertex v_x to v_y is a Φ (sequential, concurrent-synchronized, or concurrent-single) arc and the transition probability is p.*

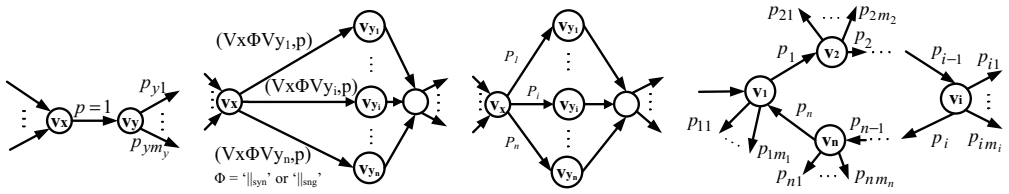


Fig. 1. Basic Composition Patterns

Workflow patterns have been discussed and defined in [9], among which we identify four patterns that directly relevant to service composition: sequential, parallel, conditional, and loop patterns (see Figure 1). Depending on the join method of the branches, parallel patterns can be further classified into two subcategories: parallel-synchronized-merge and parallel-single-merge. The definitions of the composition patterns are as follows:

Definition 2. *Sequential Pattern (see Figure 1(a)): In* $G = (V, A)$, $G' = (V', A')$ *is a Sequential Pattern where* $V' = \{v_x, v_y\}$ *and* $A' = \{(v_x - v_y, 1)\}$.

Definition 3. *Parallel Pattern (see Figure 1(b)): In* $G = (V, A)$, $G' = (V', A')$ *is a Parallel Pattern with n concurrently executed vertices where* $V' = \{v_{y_i} | i \in [1, n]\}$ *and* $A' = \{(v_x || _{syn} v_{y_i}, p) | i \in [1, n]\}$ *or* $A' = \{(v_x || _{sng} v_{y_i}, p) | i \in [1, n]\}$. *If* $A' = \{(v_x || _{syn} v_{y_i}, p) | i \in [1, n]\}$, $G' = (V', A')$ *is a Parallel-synchronized-merge Pattern. If* $A' = \{(v_x || _{sng} v_{y_i}, p) | i \in [1, n]\}$, $G' = (V', A')$ *is a Parallel-single-merge Pattern.*

In a Parallel Pattern with n concurrently running branches, if the branches join to a Web service in a synchronized mode, i.e. the Web service will only be invoked when all branches are finished, then it is a Parallel-synchronized-merge Pattern. If the invocation of the Web service is triggered by the first finished branch and other branches are ignored, then it is a Parallel-single-merge Pattern.

Definition 4. *Conditional Pattern (see Figure 1(c)): In* $G = (V, A)$, $G' = (V', A')$ *is a Conditional Pattern with n exclusively executed vertices where* $V' = \{v_{y_i} | i \in [1, n]\}$ *and* $A' = \{(v_x - v_{y_i}, p_i) | i \in [1, n]\}$.

In a Conditional Pattern with n branches, each branch has an execution probability associated with it and only one of them can be executed at a time.

Definition 5. *Loop Pattern (see Figure 1(d)): In* $G = (V, A)$, $G' = (V', A')$ *is a Loop Pattern with n vertices in the Loop where* $V' = \{v_i | i \in [1, n]\}$ *and* $A' = \{(v_i - v_{i+1}, p_i), (v_n - v_1, p_n) | i \in [1, n-1]\}$.

The n Web services in a Loop Pattern can be executed repeatedly and the loop can be left from any one of the n services.

4 QoS Aggregation

In this section, we will discuss the necessary aggregation operations used in QoS calculation for different QoS metrics, the formulas for the aggregation operations and for the composition patterns.

4.1 QoS Aggregation Operations

Composite QoS calculation depends on the composition pattern and the characteristics of the QoS metrics. For example, the execution time of a Sequential Pattern is the sum of the execution time of its component services; while the execution time of a Parallel-synchronized-merge Pattern should take the maximum execution time of the component services and the execution time of a Parallel-single-merge Pattern is the minimum execution time of the component services.

Taking composition patterns and QoS metrics into account, we define the following operations for QoS aggregation (see Table 1):

- **QoSSum**(denoted as $\circledast$): operates on the component QoS distributions by taking into consideration of the addition of their QoS values;
- **QoSMin**(denoted as $\circledwedge$): operates on the component QoS distributions by taking into consideration of the minimum of their QoS values;
- **QoSMax**(denoted as $\circledvee$): operates on the component QoS distributions by taking into consideration of the maximum of their QoS values;
- **QoSWeightedSum**(denoted as $\odot$): operates on the component QoS distributions by taking into consideration of the addition of their QoS values with path probabilities as weights. It is mainly used in the Conditional and Loop Patterns.

These operations and their relationships with composition patterns and QoS metrics (i.e. cost, time, and throughput) are summarized in Table 1.

Table 1. Operations for QoS Aggregation

QoS \ Pattern	Sequential	Para-Syn	Para-Sng	Loop	Conditional
C	QoSSum	QoSSum	QoSSum	QoSSum& QoSWeightedSum	QoSWeightedSum
T	QoSSum	QoSMax	QoSMin	QoSSum& QoSWeightedSum	QoSWeightedSum
TH	QoSMin	QoSMin	QoSMin	QoSMin& QoSWeightedSum	QoSWeightedSum

C: cost. TH: throughput. T: time related QoS metrics, such as execution time.

4.2 Formulas for QoS Aggregation Operations

In this subsection, formulas are developed for the aggregation operations defined in the previous section.

First, we introduce the following naming conventions:

- q is a variable representing a QoS metric;
- $f(q)$ denotes the density function of the probability distribution (**PDF**);
- $F(q)$ denotes the cumulative distribution function (**CDF**); $F(q)$ and $f(q)$ has the following cumulative relationship: $F(q) = \int_{-\infty}^{q} f(x)dx$ for continuous distributions or $F(q) = \sum_{q_i <= q} f(q_i)$ for discrete distributions.

It should be noted that although the discussion is based on distributions, the developed formulas for QoS aggregation are also applied to single values. This is because single values can also be represented as distributions with the help of Dirac delta function[1]. For example, if the cost of a Web service is M, then $f(q) = \delta(q - M)$. If the cost of one Web service is N_1 with a probability of p_1 and N_2 with a probability of p_2 ($p_1 + p_2 = 1$), then the distribution of this Web service can be expressed as $f(q) = p_1\delta(q - N_1) + p_2\delta(q - N_2)$.

[1] $\delta(x)$ is the Dirac delta function. $\delta(x) = +\infty$ when $x = 0$ and $\delta(x) = 0$ when $x \neq 0$.

QoSSum. Computing the PDF of the `QoSSum` of two component QoS distributions is a problem of deducing the PDF of the sum of independent variables, which is the convolution of each of their density functions [7],

$$f(q) = f_1(q) \circledast f_2(q) = \int_{\eta=0}^{q} f_1(\eta) f_2(q-\eta) d\eta \tag{1}$$

where $f(q)$ is the PDF of the QoS of a composition pattern, $f_1(q)$ and $f_2(q)$ are the PDFs of the component services.

Let us take a simple example of `execution time` for a Sequential Pattern. Let the PDFs of the execution time of the two vertices be $f_1(t)$ and $f_2(t)$ respectively. The probability for the execution time of the first one being τ ($\tau \in (0, t)$) and the second one being $t-\tau$ ($t \in (0, +\infty)$) is $f_1(\tau) f_2(t-\tau)$. Therefore, the probability for the Sequential Pattern being finished at time t is the integral of $f_1(\tau) f_2(t-\tau)$ over $(0, t)$ where τ is the variable, i.e. $f(t) = \int_{\tau=0}^{t} f_1(\tau) f_2(t-\tau) d\tau$. The result is the same as what we get from Formula (1).

QoSMin. The probability distribution of the `QoSMin` of n component QoS distributions is the distribution of the minimum of n independent variables which can be calculated as [7]:

$$F(q) = F_1(q) \circledcirc \ldots \circledcirc F_i(q) \circledcirc \ldots \circledcirc F_n(q) = 1 - \prod_{i=1}^{n} [1 - F_i(q)] \tag{2}$$

where $F(q)$ is the CDF of the QoS of a composition pattern; n is the number of component services within this pattern; and $F_i(q)$ is the CDF of the QoS of the component service i.

Then the PDF can be obtained by differentiating both sides of Equation (2) with respect to q:

$$f(q) = f_1(q) \circledcirc \ldots \circledcirc f_i(q) \circledcirc \ldots \circledcirc f_n(q) = \sum_{i=1}^{n} f_i(q) \prod_{j=1,\ldots,n \& j \neq i} [1 - F_j(q)] \tag{3}$$

where $f(q)$ is the PDF of a composition pattern; n is the number of component services within this pattern; $f_i(q)$ is the PDF of the component service i; and $F_j(q)$ is the CDF of the component service j.

Let us take a QoS metric, `throughput` as an example. Assume that X and Y are two Web services in a Sequential Pattern. The probability for them having the capability of processing up to q number of requests are $F_X(q)$ and $F_Y(q)$. The probability for none of them being able to process q number of requests is $(1 - F_X(q))(1 - F_Y(q))$, therefore, the probability for either of them being able to process q number of requests is $1 - (1 - F_X(q))(1 - F_Y(q))$. The fact that either of them is able to process q number of requests means that q is the smaller throughput of the two.

QoSMax. The distribution of the `QoSMax` of n component QoS distributions is the distribution of the maximum of n independent variables which can be calculated as [7]:

$$F(q) = F_1(q) \circledcirc \ldots \circledcirc F_i(q) \circledcirc \ldots \circledcirc F_n(q) = \prod_{i=1}^{n} F_i(q) \tag{4}$$

where $F(q)$ is the CDF of the QoS of a composition pattern; n is the number of component services within this pattern; and $F_i(q)$ is the CDF of the QoS of the component service i.

The PDF can be obtained by differentiating both sides of Equation (4) with respect to q:

$$f_(q) = f_1(q) \circledcirc \ldots \circledcirc f_i(q) \circledcirc \ldots \circledcirc f_n(q) = \sum_{i=1}^{n} f_i(q) \prod_{j=1,\ldots,n \& j \neq i} F_j(q) \tag{5}$$

where $f(q)$ is the PDF of a composition pattern; n is the number of component services within this pattern; $f_i(q)$ is the PDF of the component service i; and $F_j(q)$ is the CDF of the component service j.

Let us take `execution time` as an example. Assume that X and Y are two concurrently running Web services. The probability for X and Y to be finished within time t is $F_X(t)$ and $F_Y(t)$ respectively. Therefore, the probability for both of them to be finished within time t is $F_X(t)F_Y(t)$. The fact that both Web services can be finished within t means that t is the longer execution time of the two.

QoSWeightedSum. The QoS distribution for the `QoSWeightedSum` of component QoS distributions can be calculated as

$$f(q) = f_1(q) \odot \ldots \odot f_i(q) \odot \ldots \odot f_n(q) = \sum_{i=1}^{n} p_i f_i(q) \tag{6}$$

where $f(q)$ is the PDF of a composition pattern; n is the number of component services within this pattern; $f_i(q)$ is the PDF of the component service i; and p_i is the execution probability for the component service i.

Here we take `execution time` as an example. Assume X and Y are two Web services within a Conditional Pattern with the execution probabilities being p_1 and p_2 respectively. The probabilities for X and Y to be finished at time t are $f_X(t)$ and $f_Y(t)$ respectively. Therefore, the probability for the path of X to be finished at time t is $p_1 f_X(t)$ and for the path of Y to be finished at time t is $p_2 f_Y(t)$. Therefore, the probability for the Conditional Pattern to be finished at time t is $f(t) = p_1 f_X(t) + p_2 f_Y(t)$.

4.3 QoS Probability Distribution Aggregation for Composition Patterns

So far, we have discussed the operations and formulas involved in computing composite QoS distributions. In this section, we will explain how component

QoS distributions are aggregated for different composition patterns. Here, QoS metrics `cost`, `time`, and `throughput` will be discussed.

For a composition pattern with two component services, assume the probability distribution of composite QoS is $c(q)$ for `cost`, $t(q)$ for `time`, and $th(q)$ for `throughput`. The probability distribution of two component QoSs are $c_1(q)$ and $c_2(q)$ for `cost`, $t_1(q)$ and $t_2(q)$ for `time`, $th_1(q)$ and $th_2(q)$ for `throughput`. It can be seen from Table 1 that

- in a **Sequential Pattern**:
  $c(q) = c_1(q) \circledast c_2(q)$; $t(q) = t_1(q) \circledast t_2(q)$; $th(q) = th_1(q) \circledcirc th_2(q)$.
- in a **Parallel-synchronized-merge Pattern**:
  $c(q) = c_1(q) \circledast c_2(q)$; $t(q) = t_1(q) \circledvee t_2(q)$; $th(q) = th_1(q) \circledcirc th_2(q)$.
- in a **Parallel-single-merge Pattern**:
  $c(q) = c_1(q) \circledast c_2(q)$; $t(q) = t_1(q) \circledcirc t_2(q)$; $th(q) = th_1(q) \circledcirc th_2(q)$.
- in a **Conditional Pattern**:
  $c(q) = c_1(q) \odot c_2(q)$; $t(q) = t_1(q) \odot t_2(q)$; $th(q) = th_1(q) \odot th_2(q)$.

The QoS computation for a **Loop Pattern** is more complicated than other patterns. In [11], we have given detailed discussion on the structure analysis method for an arbitrary Loop Pattern to compute its QoS. To sum up the method in [11], statistically, a Loop Pattern can be seen as a Conditional Pattern with a Sequential Pattern in each path. With the formula for calculating the execution probability of each path of the Conditional Pattern given in [11] (see Formula (7)) and the formulas of computing the QoS of a Sequential Pattern and Conditional Pattern known (formulas (1), (3) and (6)), the distribution of the QoS of a Loop Pattern can be computed.

$$p_{path_{li}} = (\prod_{k=1}^{n} p_k)^l (\prod_{k=0}^{i-1} p_k)(1 - p_i) \tag{7}$$

where p_k is the transition probability from vertex v_k to v_{k+1} and $p_k = 1$ when $k = 0$, l is the number of times that the Loop is executed, n is the number of vertices in the Loop, and i is the index of the vertex which the Loop is jumped out of.

To compute the QoS distribution for a Loop Pattern, we can set a threshold value, TH, for $p_{path_{li}}$. Whenever $p_{path_{li}} < TH$, the iteration can be stopped. It means that $l = L$ times of looping is enough if L satisfies $(\prod_{k=1}^{n} p_k)^L (\prod_{k=0}^{i-1} p_k)(1 - p_i) < TH$.

After the QoS distributions for a Loop Pattern are computed, the Loop Pattern can be replaced by one vertex with the computed QoS distributions. Unlike other patterns, the transition probability for each outgoing arc of a Loop Pattern ($p_{i1}, \ldots, p_{im_i}$ where $i = 1, \ldots, n$ in Figure 1(d)) has to be changed accordingly. Detailed formula in calculating the probability of the outgoing arcs can be found in [11].

5 The Process of QoS Aggregation

In this section, we shall explain the complete process for calculating the QoS of a composite service.

As depicted in Figure 2, the input of the process is the service graph representing the composite service and the QoS probability distributions of component services. The output of the process is QoS probability distributions of the composite service. QoS probability distributions are stored in and retrieved from the QoS database shown in Figure 2.

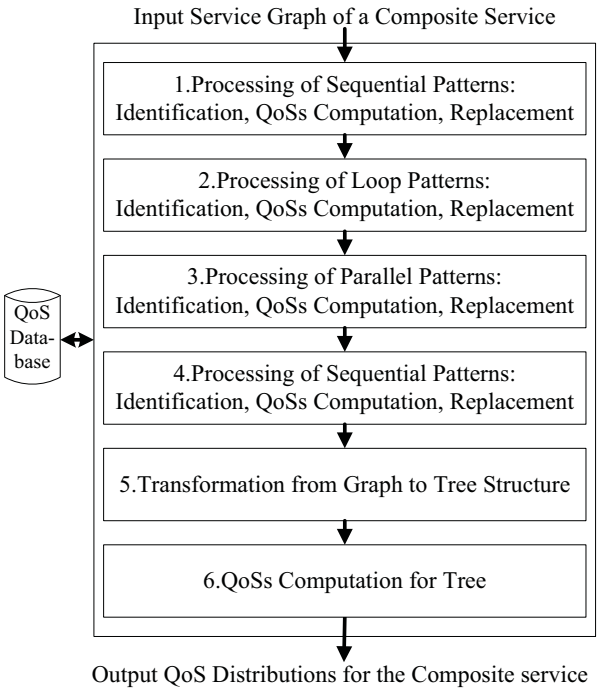


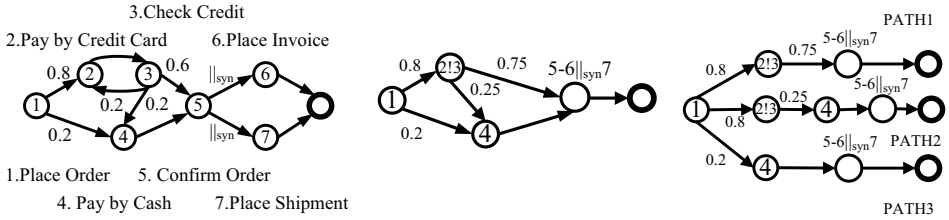
Fig. 2. Process of QoS Analysis

There are six steps in the QoS aggregation process as illustrated in Figure 2[2]. In the first three steps, Sequential Patterns, Loop Patterns, and Parallel Patterns are identified, their QoS probability distributions are computed, and they are replaced with single vertices with the computed QoSs. The identification of these patterns is based on the transition probability and connection method of arcs specified in Definition 2 to 5. The details of QoSs computation for these patterns have been provided in Section 4.3. After step 3, only Sequential Patterns and Conditional Patterns are left in the service graph. Step 4 deals with the left over Sequential Patterns as the result of the replacement of the previously processed patterns. After step 4, only Conditional Patterns are left in the service graph. Step 5 transforms the service graph into a tree structure. Step 6 calculates the QoS probability distributions of the composite service. In this step, the execution

[2] Algorithms of identifying different composition patterns out from a service graph and transforming from graph to tree can be found in [11].

probability and QoSs of each path of the tree are computed first, then the QoS probability distributions are calculated based on Formula (6).

Here is an example showing how the proposed QoS analysis method works. The QoS metric execution time will be calculated in this example.



(a) Initial Graph (b) Graph with only Conditional Patterns (c) Tree

Fig. 3. Service Graph Transformations

Figure 3(a) is a service graph for a composite service *OrderGoods*. After the service *Place Order*, a customer will either use the service *Pay by Cash* or the service *Pay by Credit Card*. If the credit card information is not approved by the service *Check Credit*, the customer may still choose the service *Pay by Credit Card*, or change to the service *Pay by Cash*. After payment is finished, the order will be confirmed by the service *Confirm Order*. Then both service *Place Invoice* and service *Place shipment* will be invoked, and the process will terminate after both of the services are finished.

The service graph in Figure 3(a) together with the probability distributions of the execution time of the seven component services are the input of the QoS analysis process. As there is no Sequential Pattern that needs to be handled immediately, the service graph remains unchanged after Step 1. In Step 2, the response time of the Loop Pattern composed of vertex 2 and 3 is calculated based on formulas (7), (1), and (6). Then this Loop Pattern is replaced by a vertex marked as 2!3 (see Figure 3(b)). In Step 3, the response time of the Parallel-synchronized-merge Pattern composed of vertex 6 and 7 is calculated based on Formula (5). Then, this Parallel-synchronized-merge Pattern is replaced by a vertex marked as $6||_{syn}7$. In Step 4, a Sequential Pattern composed of vertex 5 and vertex $6||_{syn}7$ is found. The execution time is calculated based on Formula (1). Then, this Sequential Pattern is replaced by a vertex marked as $5\text{-}6||_{syn}7$. It can be seen that after the first four steps, Figure 3(a) is changed into Figure 3(b) with only Conditional Patterns left. In Figure 3(b), both vertex 4 and vertex $5\text{-}6||_{syn}7$ are split from vertex 2!3, then vertex 4 joins vertex $5\text{-}6||_{syn}7$. It is impossible to replace this pattern with one vertex. According to step 5, the service graph with Conditional Patterns is transformed into a tree structure with three paths (see Figure 3(c)). In step 6, the probability and response time for each path of the tree structure are computed. The probability distribution of

the response time of the composite service is calculated as the `QoSWeightedSum` of the probability distribution of the response time of each path in the tree.

6 Experiment

In order to show the effectiveness of the proposed QoS analysis method, experiment is done in this section.

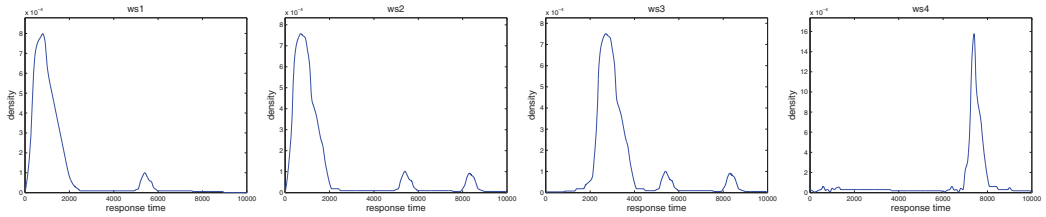
The service composition in Figure 3(a) is the example scenario. Seven Web services as depicted in Figure 3(a) are developed and deployed on Apache Tomcat 5.5 Server. The probability distributions of execution time of these seven component services are shown in Figure 4. A BPEL process is developed and deployed on Active BPEL engine for the implementation of the composite service (see Figure 3(a)).

Now let us look at how to let a Web service execution time follow a specific distribution, i.e. the execution time of a Web service per invocation must be a random value conforming to this distribution. A large array of random numbers conforming to the specified distribution is generated and stored in a file. The size of the array is set to 10000. For each execution, the Web service will randomly read one value in the file and suspend for the same amount of time as the obtained random value before it sends out a response. By doing so, a Web service conforming to certain execution time distribution is developed.

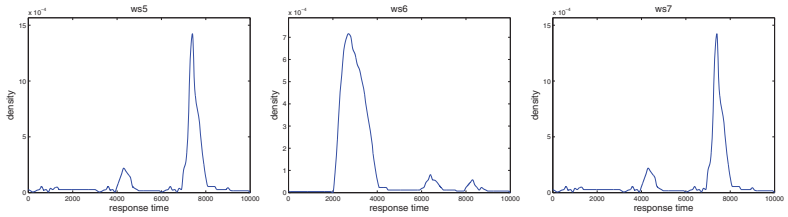
To simulate the transition probabilities in *OrderGoods*, a random number generator conforming to a uniform distribution is adopted. For each invocation of the service *Place Order*, the generated random number is compared with 0.8. If it is smaller than 0.8, the output payment method is "Credit Card"; otherwise, the output is "Paypal". Then, either of its two successors will be invoked depending on the output payment method. By doing this, the transition probabilities of its following branches are ensured to be the same as they are indicated in Figure 3(a). It is the same for developing the service *Check Credit*: if the generated random number is smaller than 0.6, the output is "Approved"; otherwise, it is "Disapproved".

The composite service is invoked for 5,000 times. For each invocation, the execution time of the composite service is recorded. The histogram of experimental result is shown in Figure 5. By using the proposed QoS analysis method, the distribution of the execution time for the composite service can be computed and the calculation result is the curve in Figure 5.

If the QoSs of component Web services are modeled as mean values (indicated in Figure 4), the QoS of the service composition in Figure 3(a) can be calculated and the value is 20942.7. The mean of the distribution calculated by our approach (22523mm) is almost the same as that of the experimental result (22706mm). Besides, it can be seen from Figure 5 that the estimation result calculated by the proposed approach fits the experimental sample very well. This experimental result provides a good indication for the effectiveness and accuracy of our proposed approach.



(a) WS1 (mean = 1360.3) (b) WS2 (mean = 1681.2) (c) WS3 (mean = 3386.0) (d) WS4 (mean = 6784.3)



(e) WS5 (mean = 6580.9) (f) WS6 (mean = 3387.6) (g) WS7 (mean = 6580.9)

Fig. 4. Distributions of Component Services

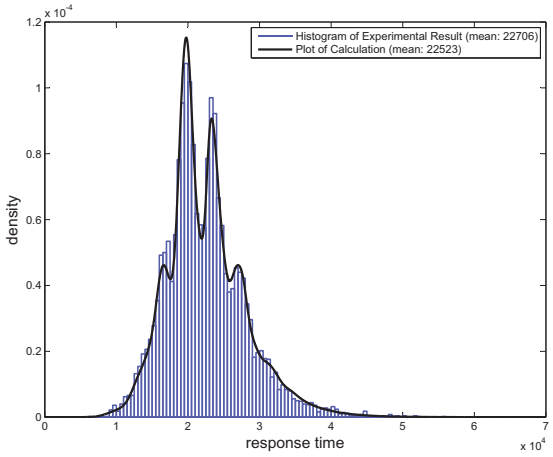


Fig. 5. Calculated QoS Distribution and Histogram of Experimental Result

7 Conclusion

Quality analysis and prediction for composite services is an important and challenging issue in distributed computing. Existing approaches either use discrete values or standard statistical distributions to represent component QoS. QoS calculation for composite service either uses aggregation approach based on single values or simulation based on standard statistical models. However in reality, the QoS of a service can be of any distribution, and may not necessarily be well represented by any existing standard statistical distribution. Any QoS calculation or simulation based on inaccurate model will not produce an effective QoS prediction for composite service. Thus, we propose a comprehensive framework for analyzing QoS of a composite service based on the probability distributions

of component QoS. A set of operations and formulas are developed to calculate the probability distribution of QoS for the four basic composition patterns. Experiment results indicate the effectiveness of our proposed method for QoS aggregation.

References

1. Qos for web services: Requirements and possible approaches. Technical Report 25, W3C Working Group (November 2003)
2. Ardagna, D., Pernici, B.: Adaptive service composition in flexible processes. IEEE Transactions on Software Engineering 33(6), 369–384 (2007)
3. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: Qos-aware replanning of composite web services. In: Proceedings of the IEEE International Conference on Web Services (ICWS), USA (2005)
4. Cardoso, J., Miller, J., Sheth, A., Arnold, J.: Quality of service for workflows and web service processes. Journal of Web Semantics 1, 281–308 (2004)
5. Hughes, C., Hillman, J.: Qos explorer: A tool for exploring qos in composed services. In: IEEE International Conference on Web Services (ICWS), USA (2006)
6. Hwang, S.-Y., Wang, H., Tang, J., Srivastava, J.: A probabilistic approach to modeling and estimating the qos of web-services-based workflows. Information Sciences 177(23), 5484–5503 (2007)
7. Papoulis, A.: Probability, random variables, and stochastic processes. McGraw-Hill, New York (1965)
8. Rosario, S., Benveniste, A., Haar, S., Jard, C.: Probabilistic qos and soft contracts for transaction-based web services orchestrations. IEEE Transactions on Services Computing 1(4), 187–200 (2008)
9. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and Parallel Databases 14(1), 5–51 (2003)
10. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Transactions on Software Engineering 30(5), 311–327 (2004)
11. Zheng, H., Zhao, W., Yang, J., Bouguettaya, A.: Qos analysis for web service composition. In: IEEE International Conference on Services Computing (SCC), India (2009)